



PATENT

- 1 -

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	:	Before the Examiner:
Balaram Sinharoy et al.	:	Li, Aimee J.
Serial No.: 09/548,469	:	Group Art Unit: 2183
Filed: April 13, 2000	:	
	:	IBM Corporation
Title: USE OF SOFTWARE HINT FOR	:	Intellectual Property Law
BRANCH PREDICTION IN THE	:	11400 Burnet Road
ABSENCE OF HINT BIT IN THE	:	Austin, Texas 78758
BRANCH INSTRUCTION	:	

**AMENDED SUPPLEMENTAL APPEAL BRIEF**

Mail Stop Appeal Brief-Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

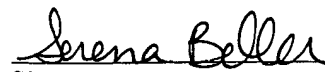
I. **REAL PARTY IN INTEREST**

The real party in interest is International Business Machines Corporation, which is the assignee of the entire right, title and interest in the above-identified patent application.

---

**CERTIFICATION UNDER 37 C.F.R. § 1.8**

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450, on April 6, 2005.

  
\_\_\_\_\_  
Signature

Serena Beller  
\_\_\_\_\_  
(Printed name of person certifying)

## II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

## III. STATUS OF CLAIMS

Claims 1-14, 39 and 40 are pending in the Application. Claims 1-14, 39 and 40 stand rejected. Claims 1-14, 39 and 40 are appealed.

## IV. STATUS OF AMENDMENTS

Appellants' response to the Office Action having the mailing date of September 25, 2003, has been considered, but the Examiner indicated that it did not place the application in condition for allowance because Appellants' arguments were deemed unpersuasive.

## V. SUMMARY OF CLAIMED SUBJECT MATTER

In one embodiment of the present invention, a method for predicting a result of a conditional branch instruction may comprise the step of determining if a specified condition register field is used to store a branch condition of the conditional branch instruction. Specification, page 17, lines 14-20; Specification, page 19, lines 9-12; Specification, page 21, lines 5-11; Specification, page 25, claim 1, lines 1-4; Figure 3A-1, steps 304, 305; Figure 3A-1, steps 307, 308; Figure 3B, steps 321, 322. The method may further comprise the step of providing a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch instruction. Specification, page 22, lines 4-17; Specification, page 25, claim 1, lines 5-7; Figure 4, steps 402, 403, 404, 406.

In another embodiment of the present invention, a processor comprises an instruction fetch unit for fetching a conditional branch instruction. Specification, page 12, line 19 – page 13, line 4; Specification, page 27, claim 8, lines 1-2; Figure 2, element 17. The processor may further comprise circuitry for determining if a specified condition register field is used to store a branch condition of the conditional branch instruction. Specification, page 17, lines 14-20; Specification, page 19, lines 9-12; Specification, page 21, lines 5-11; Specification, page 27, claim 8, lines 3-4; Figure 3A-1, steps 304, 305; Figure 3A-1, steps 307, 308; Figure 3B, steps 321, 322. The processor may further comprise circuitry for providing a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch instruction. Specification, page 22, lines 4-17; Specification, page 27, claim 8, lines 5-7; Figure 4, steps 402, 403, 404, 406.

In another embodiment of the present invention, a data processing system for predicting whether a conditional branch instruction will be taken or not taken, the data processing system may comprise the program step of determining if the conditional branch instruction is positioned at a specified address in a sequence of instructions being executed. Specification, page 19, line 16 – page 21, line 4; Specification, page 22, lines 13-17; Specification, page 41, lines 1-4; Figure 3A-1, step 314; Figure 3A-2, step 315; Figure 3B, step 323; Figure 4, step 404. The data process system may further comprise the program step of predicting whether the conditional branch instruction will be taken or not taken as a function of the position of the specified address. Specification, page 19, lines 16-18; Specification, page 22, lines 9-17; Specification, page 41, lines 5-6; Figure 4, steps 403-406.

VI. GROUND OF REJECTION TO BE REVIEWED ON APPEAL

Claims 39 and 40 stand rejected under 35 U.S.C. §102(b) as being anticipated by Hennessy's Computer Architecture: A Quantitative Approach Second Edition © 1996 (hereinafter "Hennessy"). Claims 1-14 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Henry et al. (U.S. Patent No. 6,550,004) (hereinafter "Henry") in view of Tanenbaum's Structured Computer Organization Second Edition © 1984 (hereinafter "Tanenbaum").

VII. ARGUMENTA. Claims 39 and 40 are not properly rejected under 35 U.S.C. §102(b).

The Examiner has rejected claims 39 and 40 under 35 U.S.C. §102(b) as being anticipated by Hennessy. Paper No. 13, page 2. Appellants respectfully traverse and assert that claims 39 and 40 are not properly rejected under 35 U.S.C. §102(b) as being anticipated by Hennessy for at least the reasons stated below.

For a claim to be anticipated under 35 U.S.C. §102, each and every claim limitation must be found within the cited prior art reference and arranged as required by the claim. M.P.E.P. §2131.

Appellants respectfully assert that Hennessy does not disclose "determining if the conditional branch instruction is positioned at a specified address in a sequence of instructions being executed" as recited in claim 39. The Examiner cites page 273, paragraph 2; Figure 4.22; page 274, paragraph 1; and Figure 4.23 of Hennessy as disclosing the above-cited claim limitation. Paper No. 13, page 3. Appellants respectfully traverse and assert that Hennessy instead discloses that the PC of the instruction being fetched is matched against a set of instruction addresses stored in the first column of a buffer. Hennessy further discloses that these instruction addresses represent the addresses of known branches. Hennessy further discloses that

if the PC matches one of these entries, then the instruction being fetched is a taken branch, and the second column in the buffer, predicted PC, contains the prediction for the next PC after the branch. Hennessy further discloses that fetching begins immediately at that address. Hence, Hennessy discloses matching the instruction fetched with a set of stored instruction addresses. This is not the same as determining if an instruction is positioned at a specified address in a sequence of instructions. Neither is there language in the cited passages and figures that discloses determining if a conditional branch instruction is positioned at a specified address in a sequence of instructions being executed. Thus, Hennessy does not disclose all of the limitations of claim 39, and thus Hennessy does not anticipate claim 39. M.P.E.P. §2131.

Appellants further assert that Hennessy does not disclose "predicting whether the conditional branch instruction will be taken or not taken as a function of the position of the specified address" as recited in claim 39. The Examiner cites page 273, paragraph 2; Figure 4.22; page 274, paragraph 1; and Figure 4.23 of Hennessy as disclosing the above-cited claim limitation. Paper No. 13, page 3. Appellants respectfully traverse. As stated above, Hennessy instead discloses that the PC of the instruction being fetched is matched against a set of instruction addresses stored in the first column of a buffer. Hennessy further discloses that these instruction addresses represent the addresses of known branches. Hennessy further discloses that if the PC matches one of these entries, then the instruction being fetched is a taken branch, and the second column in the buffer, predicted PC, contains the prediction for the next PC after the branch. Hennessy further discloses that fetching begins immediately at that address. Hence, Hennessy discloses matching the instruction fetched with a set of stored instruction addresses. This is not the same as predicting whether a conditional branch instruction will be taken or not taken as a function of the position of the specified address. Thus, Hennessy does not disclose all of the limitations of claim 39, and thus Hennessy does not anticipate claim 39. M.P.E.P. §2131.

Claim 40 recites combinations of features including the above combinations, and thus is not anticipated for at least the above-stated reasons. Claim 40 recites additional features which, in combination with the features of the claim upon which it depends, is not anticipated by Hennessy.

For example, Hennessy does not disclose "wherein the predicting program step will predict taken if the specified address is a multiple of specified number N" as recited in claim 40. The Examiner cites page 273, paragraph 2; Figure 4.22; page 274, paragraph 1; and Figure 4.23 of Hennessy as disclosing the above-cited claim limitation. Paper No. 13, page 3. The Examiner further states:

In regards to Hennessy, every address is a multiple of itself.  
For example, if the branch instruction is at specified address  
20, then the address is a multiple of a specified number 20.  
Paper No. 13, page 3.

Appellants respectfully traverse. As stated above, Hennessy instead discloses that the PC of the instruction being fetched is matched against a set of instruction addresses stored in the first column of a buffer. Hennessy further discloses that these instruction addresses represent the addresses of known branches. Hennessy further discloses that if the PC matches one of these entries, then the instruction being fetched is a taken branch, and the second column in the buffer, predicted PC, contains the prediction for the next PC after the branch. Hennessy further discloses that fetching begins immediately at that address. Hence, Hennessy discloses matching the instruction fetched with a set of stored instruction addresses. This is not the same as predicting taken if the specified address in the sequence of instructions being executed is a multiple of specified number N. Thus, Hennessy does not disclose all of the limitations of claim 40, and thus Hennessy does not anticipate claim 40. M.P.E.P. §2131.

Furthermore, as stated above, Hennessy does not disclose determining if a conditional branch instruction is positioned at a specified address for at least the reasons stated above. Hence, Hennessy does not disclose predicting taken if the specified address is a multiple of specified number N. Thus, Hennessy does not disclose all of the limitations of claim 40, and thus Hennessy does not anticipate claim 40. M.P.E.P. §2131.

Furthermore, the Examiner's example does not relate to predicting the conditional branch instruction taken if the specified address in the sequence of instructions being executed is a multiple of specified number N. Hence, Hennessy does not disclose all of the limitations of claim 40, and thus Hennessy does not anticipate claim 40. M.P.E.P. §2131.

As a result of the foregoing, Appellants respectfully assert that not each and every claim limitation was found within Hennessy, and thus claims 39-40 are not anticipated by Hennessy.

B. Claims 1-14 are not properly rejected under 35 U.S.C. §103(a) as being unpatentable over Henry in view of Tanenbaum.

1. The Examiner has not provided any objective evidence for modifying Henry to provide a software branch prediction.

A *prima facie* showing of obviousness requires the Examiner to establish, *inter alia*, that the prior art references teach or suggest, either alone or in combination, all of the limitations of the claimed invention, and the Examiner must provide a motivation or suggestion to combine or modify the prior art reference to make the claimed inventions. M.P.E.P. §2142. The showings must be clear and particular and supported by objective evidence. *In re Lee*, 277 F.3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1433-34 (Fed. Cir. 2002); *In re Kotzab*, 217 F.3d 1365, 1370,

55 U.S.P.Q.2d 1313, 1317 (Fed. Cir. 2000); *In re Dembiczak*, 50 U.S.P.Q.2d. 1614, 1617 (Fed. Cir. 1999). Broad conclusory statements regarding the teaching of multiple references, standing alone, are not evidence. *Id.*

The Examiner's motivation for modifying Henry with Tanenbaum to provide a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch instruction, as recited in claim 1 and similarly in claim 8, is that "the branch prediction method taught in Henry can be done in both hardware and software and it is more a design decision whether to implement the method in hardware and software." Paper No. 13, pages 5 and 6. The Examiner's motivation is insufficient to support a *prima facie* case of obviousness for at least the reasons stated below.

The Examiner's motivation does not address as to why one of ordinary skill in the art would modify Henry's branch predictor implemented in hardware to be implemented in software. While an instruction executed by hardware can be simulated in software, the Examiner has not provided motivation for actually modifying Henry's branch predictor implemented in hardware to be implemented in software. The mere fact that a reference can be modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the modification. *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1990); M.P.E.P. §2143.01. The Examiner is merely relying upon her own subjective opinion which is insufficient to support a *prima facie* case of obviousness. *In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002). Consequently, the Examiner's motivation is insufficient to support a *prima facie* case of obviousness for rejecting claims 1-14. *Id.*



2. The Examiner has not presented a reasonable expectation of success when modifying Henry to provide a software branch prediction.

The Examiner must present a reasonable expectation of success in modifying Henry's branch predictor implemented in hardware to be implemented in software (Examiner admits that Henry does not teach performing branch prediction by software) in order to establish a *prima facie* case of obviousness. *In re Merck & Co., Inc.*, 800 F.2d 1091, 231 U.S.P.Q. 375 (Fed. Cir. 1986); M.P.E.P. §2143.02.

Henry teaches a branch predictor that includes global and local Agree dynamical branch predictors, one of which is selected for correlation with a static branch prediction made based upon a test type of a conditional branch instruction specifying a condition upon which the branch will be taken. Abstract. Henry further teaches that the selection is made by correlating a selection prediction made by the static predictor based on the test type and an Agree prediction made by a selector history table based on the branch instruction address. Abstract. Henry further teaches that the dynamic predictors are updated if they are selected and incorrectly predicted the outcome. Abstract. Henry further teaches that the selector history table is updated if the selected dynamic predictor predicted incorrectly and the non-selected dynamic predictor predicted correctly. Abstract. Henry further teaches a hardware mechanism, as illustrated in Figure 2, for implementing the branch prediction as described above.

The Examiner has not provided any evidence as to how Henry's hardware mechanism, as illustrated in Figure 2, would be modified to be implemented in software or the feasibility of implementing the functionality of Henry's hardware mechanism in software. The Examiner must provide objective evidence as to how Henry's hardware mechanism, as illustrated in Figure 2, would be modified to be implemented in software or the feasibility of implementing the functionality of

Henry's hardware mechanism in software. M.P.E.P. §2143.02. Since the Examiner has not provided such evidence, the Examiner has not presented a reasonable expectation of success in modifying Henry's branch predictor implemented in hardware to be implemented in software in order to establish a *prima facie* case of obviousness. *In re Merck & Co., Inc.*, 800 F.2d 1091, 231 U.S.P.Q. 375 (Fed. Cir. 1986); M.P.E.P. §2143.02. Accordingly, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 1-14. M.P.E.P. §2143.02.

3. Henry and Tanenbaum, taken singly or in combination, do not teach or suggest the following claim limitations.

Appellants respectfully assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "providing a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch condition" as recited in claim 1 and similarly in claim 8. The Examiner cites lines 13-14 of the Abstract; column 4, lines 49-52; column 5, lines 8-12 and 35-38; column 9, lines 31-44 and Figure 2 of Henry as disclosing the above-cited claim limitation. Paper No. 13, page 4. Appellants respectfully traverse and assert that Henry instead teaches a static predictor, static predictor 222, that receives three inputs and predicts the outcome of conditional branch instructions based upon the three inputs. Henry further teaches that one of the three inputs comprises a conditional branch instruction test type. Henry further teaches that the test type specifies a condition upon which the branch instruction will be taken or not taken. Henry further teaches that the test type includes x86 conditional jump instruction (JCC) test types. Henry further teaches that the x86 conditional jump instruction test types include conditions based upon the carry, overflow, zero, parity and sign flags of the x86 FLAGS register. The Examiner cites Figure 3-9 and Table 3-2 on page 3-14 of the Intel Pentium™ Processor Family Developer's Manual Volume 3 (hereinafter "Intel") which teaches that conditional

jumps and subroutine calls allow a program to sense the state of the status flags and respond to them. Paper No. 13, page 8. However, responding to the state of a flag, e.g., the state of a flag (0 or logical value of 1) may be used to suppress a conditional jump, is not the same as providing a prediction based on whether a specified field, e.g., field 1, is used to store a condition. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 1 and 8, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Claims 2-7 and 9-14 recite combinations of features including the above combinations, and thus are patentable for at least the above-stated reasons. Claims 2-7 and 9-14 recites additional features which, in combination with the features of the claims upon which they depend, are patentable over Henry in view of Tanenbaum.

For example, Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is used to store the branch condition of the conditional branch instruction" as recited in claim 2 and similarly in claim 9. The Examiner cites lines 13-14 of the abstract; column 4, lines 49-52; column 5, lines 8-12 and 35-38; column 9, lines 31-44 and Figure 2 of Henry as disclosing the above-cited claim limitation. Paper No. 13, page 6. Appellants respectfully traverse. As stated above, Henry instead teaches a static predictor, static predictor 222, that receives three inputs and predicts the outcome of conditional branch instructions based upon the three inputs. Henry further teaches that one of the three inputs comprises a conditional branch instruction test type. Henry further teaches that the test type specifies a condition upon which the branch instruction will be taken or not taken. Henry further teaches that the test type includes x86 conditional jump instruction (JCC) test types. Henry further teaches that the x86 conditional jump instruction test types include conditions based upon the carry,

overflow, zero, parity and sign flags of the x86 FLAGS register. The Examiner cites Figure 3-9 and Table 3-2 on page 3-14 of Intel which teaches that conditional jumps and subroutine calls allow a program to sense the state of the status flags and respond to them. Paper No. 13, page 8. However, responding to the state of a flag, e.g., the state of a flag (0 or logical value of 1) may be used to suppress a conditional jump, is not the same as providing a prediction based on whether a specified field, e.g., field 1, is used to store a condition. Hence, Henry does not teach making a branch prediction based on whether a particular condition register field is used to store a branch condition. Neither does Henry teach predicting that a conditional branch instruction will be taken if the specified condition register field is used to store the branch condition of the conditional branch instruction. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 2 and 9, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction" as recited in claim 3 and similarly in claim 10. The Examiner cites lines 13-14 of the abstract; column 4, lines 49-52; column 5, lines 8-12 and 35-38; column 9, lines 31-44 and Figure 2 of Henry as disclosing the above-cited claim limitation. Paper No. 13, page 6. Appellants respectfully traverse. As stated above, Henry instead teaches a static predictor, static predictor 222, that receives three inputs and predicts the outcome of conditional branch instructions based upon the three inputs. Henry further teaches that one of the three inputs comprises a conditional branch instruction test type. Henry further teaches that the test type specifies a condition upon which the branch instruction will be taken or not taken. Henry further teaches that the test type includes x86 conditional jump instruction (JCC) test types. Henry

further teaches that the x86 conditional jump instruction test types include conditions based upon the carry, overflow, zero, parity and sign flags of the x86 FLAGS register. The Examiner cites Figure 3-9 and Table 3-2 on page 3-14 of Intel which teaches that conditional jumps and subroutine calls allow a program to sense the state of the status flags and respond to them. Paper No. 13, page 8. However, responding to the state of a flag, e.g., the state of a flag (0 or logical value of 1) may be used to suppress a conditional jump, is not the same as providing a prediction based on whether a specified field, e.g., field 1, is not used to store a condition. Hence, Henry does not teach making a branch prediction based on whether a particular condition register field is not used to store a branch condition. Neither does Henry teach predicting that a conditional branch instruction will not be taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 3 and 10, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is used to store the branch condition of the conditional branch instruction" as recited in claim 4 and similarly in claim 11. The Examiner cites lines 13-14 of the abstract; column 4, lines 49-52; column 5, lines 8-12 and 35-38; column 9, lines 31-44 and Figure 2 of Henry as disclosing the above-cited claim limitation. Paper No. 13, page 7. Appellants respectfully traverse. As stated above, Henry instead teaches a static predictor, static predictor 222, that receives three inputs and predicts the outcome of conditional branch instructions based upon the three inputs. Henry further teaches that one of the three inputs comprises a conditional branch instruction test type. Henry further teaches that the test type specifies a condition

upon which the branch instruction will be taken or not taken. Henry further teaches that the test type includes x86 conditional jump instruction (JCC) test types. Henry further teaches that the x86 conditional jump instruction test types include conditions based upon the carry, overflow, zero, parity and sign flags of the x86 FLAGS register. The Examiner cites Figure 3-9 and Table 3-2 on page 3-14 of Intel which teaches that conditional jumps and subroutine calls allow a program to sense the state of the status flags and respond to them. Paper No. 13, page 8. However, responding to the state of a flag, e.g., the state of a flag (0 or logical value of 1) may be used to suppress a conditional jump, is not the same as providing a prediction based on whether a specified field, e.g., field 1, is used to store a condition. Hence, Henry does not teach making a branch prediction based on whether a particular condition register field is used to store a branch condition. Neither does Henry teach predicting that a conditional branch instruction will not be taken if the specified condition register field is used to store the branch condition of the conditional branch instruction. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 4 and 11, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction" as recited in claim 5 and similarly in claim 12. The Examiner cites lines 13-14 of the abstract; column 4, lines 49-52; column 5, lines 8-12 and 35-38; column 9, lines 31-44 and Figure 2 of Henry as disclosing the above-cited claim limitation. Paper No. 13, page 7. Appellants respectfully traverse. As stated above, Henry instead teaches a static predictor, static predictor 222, that receives three inputs and predicts the outcome of conditional branch instructions based upon the three inputs. Henry

further teaches that one of the three inputs comprises a conditional branch instruction test type. Henry further teaches that the test type specifies a condition upon which the branch instruction will be taken or not taken. Henry further teaches that the test type includes x86 conditional jump instruction (JCC) test types. Henry further teaches that the x86 conditional jump instruction test types include conditions based upon the carry, overflow, zero, parity and sign flags of the x86 FLAGS register. The Examiner cites Figure 3-9 and Table 3-2 on page 3-14 of Intel which teaches that conditional jumps and subroutine calls allow a program to sense the state of the status flags and respond to them. Paper No. 13, page 8. However, responding to the state of a flag, e.g., the state of a flag (0 or logical value of 1) may be used to suppress a conditional jump, is not the same as providing a prediction based on whether a specified field, e.g., field 1, is not used to store a condition. Hence, Henry does not teach making a branch prediction based on whether a particular condition register field is not used to store a branch condition. Neither does Henry teach predicting that a conditional branch instruction will be taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 5 and 12, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the specified condition register field is N, where N is an integer" as recited in claim 6 and similarly in claim 13. Appellants further assert that Henry and Tanenbaum, taken singly or in combination, do not teach or suggest "wherein the specified condition register field is a multiple of N" as recited in claim 7 and similarly in claim 14. The Examiner cites column 7, lines 36-38 and column 9, lines 31-44 of Henry as teaching the above-cited claim limitation. Paper No. 13, page 7. Appellants respectfully traverse and assert that Henry instead teaches a register file, register file 105, that includes a status flags register that is used in

determining whether branch conditions have been satisfied. While the status flags register is used in determining whether branch conditions have been satisfied, presumably based on the state of flags stored in such a register, Henry does not teach that a prediction is a function of whether a specified field in the status flags register was used to store a branch condition. Further, Henry does not teach that such a specified field is N or a multiple of N. Therefore, the Examiner has not presented a *prima facie* case of obviousness in rejecting claims 6, 7, 13 and 14, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

As a result of the foregoing, Appellants respectfully assert that there are numerous claim limitations not taught or suggested in the cited prior art, and thus the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 1-14 as being unpatentable over Henry in view of Tanenbaum. M.P.E.P. §2143.



VIII. CONCLUSION

For at least the reasons stated above and in the Appeal Brief filed by Appellants on August 9, 2004, the rejections of claims 1-14 and 39-40 are in error. Appellants respectfully request reversal of the rejections and allowance of claims 1-14 and 39-40.

Respectfully submitted,

WINSTEAD SECHREST & MINICK P.C.

Attorneys for Appellants

By: \_\_\_\_\_

Robert A. Voigt, Jr.

Reg. No. 47,159

Kelly K. Kordzik

Reg. No. 36,571

P.O. Box 50784  
Dallas, Texas 75201  
(512) 370-2832

**APPENDIX**

1. A method for predicting a result of a conditional branch instruction, comprising the steps of:

determining if a specified condition register field is used to store a branch condition of the conditional branch instruction; and

providing a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch instruction.

2. The method as recited in claim 1, wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is used to store the branch condition of the conditional branch instruction.

3. The method as recited in claim 2, wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction.

4. The method as recited in claim 1, wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is used to store the branch condition of the conditional branch instruction.

5. The method as recited in claim 4, wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction.

6. The method as recited in claim 1, wherein the specified condition register field is N, where N is an integer.
7. The method as recited in claim 6, wherein the specified condition register field is a multiple of N.
8. A processor comprising:
  - an instruction fetch unit for fetching a conditional branch instruction;
  - circuitry for determining if a specified condition register field is used to store a branch condition of the conditional branch instruction; and
  - circuitry for providing a software branch prediction of the conditional branch instruction as a function of the determination if the specified condition register field is used to store the branch condition of the conditional branch instruction.
9. The processor as recited in claim 8, wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is used to store the branch condition of the conditional branch instruction.
10. The processor as recited in claim 9, wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction.
11. The processor as recited in claim 8, wherein the software branch prediction predicts that the conditional branch instruction will be not taken if the specified condition register field is used to store the branch condition of the conditional branch instruction.

**AT9-99-129****PATENT**

12. The processor as recited in claim 11, wherein the software branch prediction predicts that the conditional branch instruction will be taken if the specified condition register field is not used to store the branch condition of the conditional branch instruction.

13. The processor as recited in claim 8, wherein the specified condition register field is N, where N is an integer.

14. The processor as recited in claim 13, wherein the specified condition register field is a multiple of N.

39. A data processing system for predicting whether a conditional branch instruction will be taken or not taken, the data processing system comprising the program steps of:

determining if the conditional branch instruction is positioned at a specified address in a sequence of instructions being executed; and

predicting whether the conditional branch instruction will be taken or not taken as a function of the position of the specified address.

40. The data processing system as recited in claim 39, wherein the predicting program step will predict taken if the specified address is a multiple of specified number N.